WHAT WE CLAIM IS:

1. A method for operating a distributed computing system, said system including a multiplicity of network-connected worker processors and at least one supervisory processor, said supervisory processor configured to assign tasks to, and monitor the status of, said worker processors, said method comprising:

assigning tasks to a plurality of said worker processors by sending task-assignment messages, via said network, from said at least one supervisory processor to said plurality of worker processors; and,

monitoring, on a substantially continuous basis, the status of at least each of said plurality of assigned worker processors until each said processor completes its assigned task.

2. A method for operating a distributed computing system, as defined in claim 1, wherein monitoring, on a substantially continuous basis, the status of at least each of said plurality of assigned worker processors comprises receiving status messages from at least each of said plurality of assigned worker processors until each said processor completes its assigned task.

3. A method for operating a distributed computing system, as defined in claim 2, wherein monitoring, on a substantially continuous basis, the status of at least each of said plurality of worker processors further comprises detecting abnormalities in the operation of said plurality of assigned worker processors, and/or

their associated network connections, by detecting an absence of expected status message(s) received by said at least one supervisory processor.

4. A method for operating a distributed computing system, as defined in claim 3, wherein said act of detecting an absence of expected status message(s) received by said at least one supervisory processor is repeated at least once every ten minutes.

5. A method for operating a distributed computing system, as defined in claim 3, wherein said act of detecting an absence of expected status message(s) received by said at least one supervisory processor is repeated at least once every five minutes.

6. A method for operating a distributed computing system, as defined in claim 3, wherein said act of detecting an absence of expected status message(s) received by said at least one supervisory processor is repeated at least once every two minutes.

7. A method for operating a distributed computing system, as defined in claim 3, wherein said act of detecting an absence of expected status message(s) received by said at least one supervisory processor is repeated at least once each minute.

8. A method for operating a distributed computing system, as defined in claim 3, wherein said act of detecting an absence of expected status message(s) received by said at least one supervisory processor is repeated at least once every

thirty seconds.

9. A method for operating a distributed computing system, as defined in claim 3, wherein said act of detecting an absence of expected status message(s) received by said at least one supervisory processor is repeated at least once every ten seconds.

10. A method for operating a distributed computing system, as defined in claim 3, wherein said act of detecting an absence of expected status message(s) received by said at least one supervisory processor is repeated at least once every second.

11. A method for operating a distributed computing system, as defined in claim 3, wherein said act of detecting an absence of expected status message(s) received by said at least one supervisory processor is repeated at least once every tenth of a second.

12. A method for operating a distributed computing system, as defined in claim 3, wherein said act of detecting an absence of expected status message(s) received by said at least one supervisory processor is repeated at least once every hundredth of a second.

13. A method for operating a distributed computing system, as defined in claim 3, wherein said act of detecting an absence of expected status message(s) received by said at least one supervisory processor is repeated at least once each millisecond.

14. A method for operating a distributed computing system, as defined in claim 1, wherein monitoring, on a substantially continuous basis, the status of at least each of said plurality of assigned worker processors comprises:

detecting the presence of non-assigned-task-related activity on said worker processors.

15. A method for operating a distributed computing system, as defined in claim 14, wherein detecting the presence of non-assigned-task-related activity on said worker processors includes:

running an activity monitor program on each of said assigned worker processors.

16. A method for operating a distributed computing system, as defined in claim 15, wherein:

the activity monitor programs running on each of said assigned worker processors behave substantially like screen saver programs.

17. A method for operating a distributed computing system, as defined in claim 15, wherein:

the activity monitory programs running on each of said assigned worker processors send, in response to detection of keyboard activity, a message to at least one of said at least one supervisory processor(s).

18. A method for operating a distributed computing system, as defined in claim 15, wherein:

the activity monitory programs running on each of said assigned worker processors send, in response to detection of mouse activity, a message to at least one of said at least one supervisory processor(s).

19. A method for operating a distributed computing system, as defined in claim 15, wherein:

the activity monitory programs running on each of said assigned worker processors send, in response to detection of pointer activity, a message to at least one of said at least one supervisory processor(s).

20. A method for operating a distributed computing system, as defined in claim 15, wherein:

the activity monitory programs running on each of said assigned worker processors send, in response to detection of touchscreen activity, a message to at least one of said at least one supervisory processor(s).

21. A method for operating a distributed computing system, as defined in claim 15, wherein:

the activity monitory programs running on each of said assigned worker

processors send, in response to detection of voice activity, a message to at least one of said at least one supervisory processor(s).

22. A method for operating a distributed computing system, as defined in claim 15, wherein:

the activity monitory programs running on each of said assigned worker processors send, in response to detection of execution of substantial non-assigned-task-related processes, a message to at least one of said at least one supervisory processor(s).

23. A method for operating a distributed computing system, as defined in claim 14, wherein detecting the presence of non-assigned-task-related activity on said worker processors includes:

determining, in response to an activity monitor message received by at least one of said at least one supervisory of said processor(s), that at least one of said assigned worker processors is undertaking non-assigned-task-related activity.

24. A method for operating a distributed computing system, as defined in claim 23, wherein the activity monitor message is generated by an activity monitor program running on one of said assigned worker processors.

25. A method for operating an always-live distributed computing system, comprising:

providing a pool of worker processors, each having installed worker processor software, and each connected to an always-on, peer-to-peer computer network;

providing at least one supervisory processor, also connected to said always-on, peer-to-peer computer network;

using said at least one supervisory processor to monitor, on a substantially continuous basis, the status of worker processors expected to be engaged in the processing of assigned tasks; and,

using said at least one supervisory processor to reassign tasks, as needed, to achieve substantially uninterrupted processing of assigned tasks.

26. A method for operating an always-live distributed computing system, as defined in claim 25, wherein providing a pool of worker processors further includes ensuring that each of said worker processors is linked to said always-on, peer-to-peer computer network through a high-bandwidth connection.

27. A method for operating an always-live distributed computing system, as defined in claim 25, wherein providing a pool of worker processors further includes ensuring that each of said worker processors is linked to said always-on, peer-to-peer computer network at a data rate of at least 100 kilobits/sec.

28. A method for operating an always-live distributed computing system, as defined in claim 25, wherein providing a pool of worker processors further includes ensuring that each of said worker processors is linked to said always-on, peer-to-peer computer network at a data rate of at least 250 kilobits/sec.

29. A method for operating an always-live distributed computing system, as defined in claim 25, wherein providing a pool of worker processors further includes ensuring that each of said worker processors is linked to said always-on, peer-to-peer computer network at a data rate of at least 1 megabit/sec.

30. A method for operating an always-live distributed computing system, as defined in claim 25, wherein providing a pool of worker processors further includes ensuring that each of said worker processors is linked to said always-on, peer-to-peer computer network at a data rate of at least 10 megabits/sec.

31. A method for operating an always-live distributed computing system, as defined in claim 25, wherein providing a pool of worker processors further includes ensuring that each of said worker processors is linked to said always-on, peer-to-peer computer network at a data rate of at least 100 megabits/sec.

32. A method for operating an always-live distributed computing system, as defined in claim 25, wherein providing a pool of worker processors further includes ensuring that each of said worker processors is linked to said always-on, peer-to-peer computer network at a data rate of at least 1 gigabit/sec.

33. A method for operating an always-live distributed computing system, as defined in claim 25, wherein using said at least one supervisory processor to monitor the status of worker processors expected to be engaged in the processing of assigned tasks includes:

sending a status-request message to, and receiving a return acknowledgement from, each worker processor that is expected to be engaged in the processing of assigned tasks.

34. A method for operating an always-live distributed computing system, as defined in claim 33, wherein said process of sending a status-request message to, and receiving a return acknowledgement from, each worker processor that is expected to be engaged in the processing of assigned tasks is repeated at least once every second.

35. A method for operating an always-live distributed computing system, as defined in claim 33, wherein said process of sending a status-request message to, and receiving a return acknowledgement from, each worker processor that is expected to be engaged in the processing of assigned tasks is repeated at least once every tenth of a second.

36. A method for operating an always-live distributed computing system, as defined in claim 33, wherein said process of sending a status-request message to, and receiving a return acknowledgement from, each worker processor that is expected to be engaged in the processing of assigned tasks is repeated at least

once every hundredth of a second.

37. A method for operating an always-live distributed computing system, as defined in claim 33, wherein said process of sending a status-request message to, and receiving a return acknowledgement from, each worker processor that is expected to be engaged in the processing of assigned tasks is repeated at least once every millisecond.

38. A method for operating an always-live distributed computing system, as defined in claim 25, wherein using said at least one supervisory processor to monitor the status of worker processors expected to be engaged in the processing of assigned tasks includes:

periodically checking to ensure that a heartbeat message has been received, within a preselected frequency interval, from each worker processor that is expected to be engaged in the processing of assigned tasks.

39. A method for operating an always-live distributed computing system, as defined in claim 38, wherein said preselected frequency interval is less than one second.

40. A method for operating an always-live distributed computing system, as defined in claim 38, wherein said preselected frequency interval is less than one tenth of a second.

41. A method for operating an always-live distributed computing system, as defined in claim 38, wherein said preselected frequency interval is less than one hundredth of a second.

42. A method for operating an always-live distributed computing system, as defined in claim 38, wherein said preselected frequency interval is less than one millisecond.

43. A method for operating an always-live distributed computing system, as defined in claim 25, wherein using said at least one supervisory processor to reassign tasks, as needed, to achieve substantially uninterrupted processing of assigned tasks comprises:

detecting aberrant behavior among the worker processors expected to be engaged in the processing of assigned tasks; and,

assigning tasks expected to be completed by said aberrant-behaving worker processor(s) to other available processor(s) in said worker processor pool.

44. A method for operating a network-connected processor as a processing element in a distributed processing system, the method comprising:

installing software that enables said network-connected processor to receive tasks from, and provide results to, one or more independent, network-connected resource(s); and,

using the software installed on said network-connected processor to provide substantially continuous status information to an independent, network-connected resource.

45. A method for operating a network-connected processor as a processing element in a distributed processing system, as defined in claim 44, wherein using the software installed on said network-connected processor to provide substantially continuous status information to an independent, network-connected resource includes:

sending a heartbeat message to said independent, network-connected resource at least once every second.

46. A method for operating a network-connected processor as a processing element in a distributed processing system, as defined in claim 44, wherein using the software installed on said network-connected processor to provide substantially continuous status information to an independent, network-connected resource includes:

sending a heartbeat message to said independent, network-connected resource at least once every tenth of a second.

47. A method for operating a network-connected processor as a processing element in a distributed processing system, as defined in claim 44, wherein using the software installed on said network-connected processor to provide substantially continuous status information to an independent, network-connected resource

includes:

sending a heartbeat message to said independent, network-connected resource at least once every hundredth of a second.

48. A method for operating a network-connected processor as a processing element in a distributed processing system, as defined in claim 44, wherein using the software installed on said network-connected processor to provide substantially continuous status information to an independent, network-connected resource includes:

sending a heartbeat message to said independent, network-connected resource at least once every millisecond.

49. A method for operating a network-connected processor as a processing element in a distributed processing system, as defined in claim 44, wherein using the software installed on said network-connected processor to provide substantially continuous status information to an independent, network-connected resource includes:

responding to status-request messages, received from said independent, network-connected resource, within one second.

50. A method for operating a network-connected processor as a processing element in a distributed processing system, as defined in claim 44, wherein using the software installed on said network-connected processor to provide substantially continuous status information to an independent, network-connected resource

includes:

responding to status-request messages, received from said independent, network-connected resource, within one tenth of a second.

51. A method for operating a network-connected processor as a processing element in a distributed processing system, as defined in claim 44, wherein using the software installed on said network-connected processor to provide substantially continuous status information to an independent, network-connected resource includes:

responding to status-request messages, received from said independent, network-connected resource, within one hundredth of a second.

52. A method for operating a network-connected processor as a processing element in a distributed processing system, as defined in claim 44, wherein using the software installed on said network-connected processor to provide substantially continuous status information to an independent, network-connected resource includes:

responding to status-request messages, received from said independent, network-connected resource, within one millisecond.

52. A method for operating a network-connected processor as a processing

element in a distributed processing system, as defined in claim 44, wherein using the software installed on said network-connected processor to provide substantially continuous status information to an independent, network-connected resource includes:

sending, in response to a change in status of said network-connected processor, a status-update message to said independent, network-connected resource within one second.

53. A method for operating a network-connected processor as a processing element in a distributed processing system, as defined in claim 44, wherein using the software installed on said network-connected processor to provide substantially continuous status information to an independent, network-connected resource includes:

sending, in response to a change in status of said network-connected processor, a status-update message to said independent, network-connected resource within one tenth of a second.

54. A method for operating a network-connected processor as a processing element in a distributed processing system, as defined in claim 44, wherein using the software installed on said network-connected processor to provide substantially continuous status information to an independent, network-connected resource includes:

sending, in response to a change in status of said network-connected processor, a status-update message to said independent, network-connected resource within one hundredth of a second.

55. A method for operating a network-connected processor as a processing element in a distributed processing system, as defined in claim 52, wherein the change in status that initiates the sending of a status-update message is additional demand for the processing resources of the network-connected processor.

56. A method for operating a network-connected processor as a processing element in a distributed processing system, as defined in claim 52, wherein the change in status that initiates the sending of a status-update message is user input-related activity on the network-connected processor.

57. A distributed computing system comprising:

a multiplicity of worker processors;

at least one supervisory processor, configured to assign tasks to, and monitor the status of, said worker processors;

an always-on, peer-to-peer computer network linking said worker processors and said supervisory processor(s); and,

at least one of said at least one supervisory processor(s) including a monitoring module, which monitors the status of worker processors expected to be executing assigned tasks, so as to ensure that the distributed computing system maintains always-

live operation.

58. A distributed computing system, as defined in claim 57, wherein the monitoring module receives status messages from at least each of the worker processors expected to be executing assigned tasks.

59. A distributed computing system, as defined in claim 58, wherein the monitoring module detects abnormalities in the operation of said worker processors expected to be executing assigned tasks, and/or their associated network connections, by detecting an absence of expected status messages received from said worker processors.

60. A distributed computing system, as defined in claim 59, wherein the monitoring module checks for an absence of expected status messages at least once each minute.

61. A distributed computing system, as defined in claim 59, wherein the monitoring module checks for an absence of expected status messages at least once every ten seconds.

62. A distributed computing system, as defined in claim 59, wherein the monitoring module checks for an absence of expected status messages at least once each second.

63. A distributed computing system, as defined in claim 59, wherein the monitoring module checks for an absence of expected status messages at least once every tenth of a second.

64. A distributed computing system, as defined in claim 57, wherein the monitoring module detects the presence of non-assigned-task-related activity on the worker processors expected to be executing assigned tasks.

65. A distributed computing system, as defined in claim 64, further comprising:

activity monitor programs running on each of the worker processors expected to be executing assigned tasks.

66. A distributed computing system, as defined in claim 65, wherein the activity monitor programs comprise screensaver programs.

67. A distributed computing system, as defined in claim 64, wherein the activity monitor programs detect at least one of the following types of non-assigned-task-related activity:

keyboard activity;

mouse activity;

pointer activity;

touchscreen activity;

voice activity; and,

execution of substantial non-assigned-task-related processes.

68. A distributed computing system, as defined in claim 64, wherein the activity monitor programs detect at least three of the following types of non-assigned-task-related activity:

keyboard activity;

mouse activity;

pointer activity;

touchscreen activity;

voice activity; and,

execution of substantial non-assigned-task-related processes.

69. An always-live distributed computing system, comprising:

a pool of worker processors, each having installed worker processor software, and each connected to an always-on, peer-to-peer computer network; and,

at least one supervisory processor, also connected to said always-on, peer-to-peer computer network, and configured to assign tasks to said worker processors, monitor, on a substantially continuous basis, the status of worker processors expected to be engaged in the processing of assigned tasks and reassign tasks, as needed, to achieve substantially uninterrupted processing of assigned tasks.

70. An always-live distributed computing system, as defined in claim 69, wherein said computer network has a bandwidth of at least 250 kilobits/second.

71. An always-live distributed computing system, as defined in claim 69, wherein said computer network has a bandwidth of at least 1 megabit/second.

72. An always-live distributed computing system, as defined in claim 69, wherein the at least one supervisory processor monitors the status of worker processors expected to be engaged in the processing of assigned tasks by sending a status-request message to, and receiving a return acknowledgement from, each worker processor that is expected to be engaged in the processing of assigned tasks.

73. An always-live distributed computing system, as defined in claim 69, wherein the process of sending a status-request message to, and receiving a return acknowledgement from, each worker processor that is expected to be engaged in the processing of assigned tasks is repeated at least once every 10 seconds.

74. An always-live distributed computing system, as defined in claim 69, wherein the process of sending a status-request message to, and receiving a return acknowledgement from, each worker processor that is expected to be engaged in the processing of assigned tasks is repeated at least once each second.

75. An always-live distributed computing system, as defined in claim 69, wherein the process of sending a status-request message to, and receiving a return acknowledgement from, each worker processor that is expected to be engaged in the processing of assigned tasks is repeated at least twenty times each second.

76. An always-live distributed computing system, as defined in claim 69, wherein the at least one supervisory processor monitors the status of worker processors expected to be engaged in the processing of assigned tasks by

periodically checking to ensure that a heartbeat message has been received, within a preselected frequency interval, from each worker processor that is expected to be engaged in the processing of assigned tasks.

77. An always-live distributed computing system, as defined in claim 76, wherein the preselected frequency interval is less than one second.

78. An always-live distributed computing system, as defined in claim 76, wherein the preselected frequency interval is less than one tenth of a second.

79. An always-live distributed computing system, as defined in claim 76, wherein the preselected frequency interval is less than one hundredth of a second.

80. A processing element for use in a distributed processing system, the processing element comprising:

at least one processor;

memory;

at least one high-bandwidth interface to a computer network; and,

worker processor software, configured to receive tasks via said high-bandwidth interface and to provide substantially continuous status information via said high-bandwidth interface.

81. A processing element, as defined in claim 80, wherein substantially continuous status information is provided by sending periodic heartbeat messages.

82. A processing element, as defined in claim 80, wherein substantially continuous status information is provided by sending prompt responses to received

status-request messages.

83. A processing element, as defined in claim 80, wherein substantially continuous status information is provided by promptly sending a status-update message in response to a change in status.

84. Article(s)-of-manufacture for use in connection with a network-based distributed computing system, the article(s)-of-manufacture comprising at least one computer-readable medium containing instructions which, when executed, cause:

assignment of tasks to a plurality of worker processors via said network; and,

monitoring, on a substantially continuous basis, of the status of at least each of said plurality of assigned worker processors until each said processor completes its assigned task.

85. Article(s)-of-manufacture for use in connection with an always-live distributed computing system, the article(s)-of-manufacture comprising at least one computer-readable medium containing instructions which, when executed, cause:

a pool of worker processors to install worker processor software provided via an always-on, peer-to-peer computer network;

provide communication paths between said worker processors and at least one supervisory processor via said always-on, peer-to-peer computer network;

cause said at least one supervisory processor to monitor, on a substantially continuous basis, the status of worker processors expected to be engaged in the processing of assigned tasks; and,

cause said at least one supervisory processor to reassign tasks, as needed, to achieve substantially uninterrupted processing of assigned tasks.

86. Article(s)-of-manufacture for use in connection with a processing element constituting a part of a distributed computing system, the article(s)-of-manufacture comprising at least one computer-readable medium containing instructions which, when executed, cause:

worker processor software to be installed that permits said processing element to receive tasks from, and provide results to, one or more independent, network-connected resource(s); and,

said installed worker processor software to be executed and provide substantially continuous status information to one or more of said independent, network-connected resource(s).